|  |
| --- |
| FILS |
| BAMOTSU |
| BAM - Java image binarization using Otsu’s algorithm |
|  |
| **Carmen Casaru, Andrei Moisei, Mihaita Tinta** |
| **11/5/2013** |

|  |
| --- |
| There are many binarization algorithms these days and almost all of them involve several steps or processing. The raw image is transformed in a black and white version. This step is critical because many errors can be propagated from here. In character recognition is very important to separate the background from the letters. |

Contents

[1. Document Purpose 2](#_Toc371547800)

[2. Objectives 2](#_Toc371547801)

[3. Document Overview 2](#_Toc371547802)

[4. Data Design 3](#_Toc371547803)

[4.1 BAM Solution Overview 3](#_Toc371547804)

[5. Architectural Design 4](#_Toc371547805)

[5.1 System Architecture 4](#_Toc371547806)

[5.2 Architectural Patterns 5](#_Toc371547807)

[5.3 Implementation Requirements 7](#_Toc371547808)

[5.4 Components Interaction 7](#_Toc371547809)

[6. User Interface Design 8](#_Toc371547810)

[6.1 Flow Charts 8](#_Toc371547811)

[7. Testing Issues 11](#_Toc371547812)

[7.1 Critical Components. Alternatives 11](#_Toc371547813)

[8. Work Breakdown Structure Diagram 12](#_Toc371547814)

[9. Gantt Diagram 12](#_Toc371547815)

[10. Git Repository 12](#_Toc371547816)

# Document Purpose

The purpose of this document is to describe the way in which the “Voting-Based Image Binarization” software product should be implemented.

# Objectives

The main Goal of this project is to an ”Image Binarization System” (IBS), which is able to convert images into binary ones and also create an additional matrix representing the level of trust for every pixel of the initial image. The IBS will consist from two parts:

* A ”Binarization Algorithm Module” (BAM):

This will be an executable which will receive an input continuous-tone image and will produce an output binary image.

* A “Voting Binarization Algorithm Module” (VBAM): using more BAMs a “smart-voting” technology will be used to blend the independent BAMs results into a binary image.

# Document Overview

The document is structured in 10 chapters, the first offering a general description of the software product about the initial situation, the purpose of the project, the context and the benefits of the project.

The first chapter presents the document purpose.

In the second chapter a list of the objectives is detailed.

The third chapter consists of the document overview.

The fourth chapter presents the data design details.

The fifth chapter defines the architectural design of the solution.

The sixth chapter describes the user interface design

The seventh chapter details the testing issues that may be encountered within the system and offers alternatives in order to handle such problems.

The eighth chapter presents the Work Breakdown Structure diagram

The ninth chapter presents the Gantt Diagram

Last, but not least, the 10th chapter refers to the Git repository that we have set-up for this particular solution.

# Data Design

The Image Binarization software module is designed as a desktop application used to convert color images into black-and-white ones.

The solution should support multiple image types and it should also offer users the possibility to choose which format the binarized imaged is saved under.

The supported image formats supported are the following:

* Jpg
* Png
* Bmp

## 4.1 BAM Solution Overview

The BAM will be an executable which will receive as input two files (an input\_image and an output\_image). The BAM will return a status-code:

* zero for no error (results are valid and the output image will be used for voting purposes) and
* nonzero in case of an error occurrence (the error code should specify the error type; in this case the result will not be considered).

The output of the BAM is

* a 1bpp image (output\_image): the actual binarization
* an 8bpp image (output\_image- confidence): a gray-scale image containing the confidence for the binarization for every pixel
  + a “0” value means that the respective pixel was randomly assigned a color (black or white);
  + a “255” value means that the algorithm is absolutely certain that the respective color of the pixel is correctly assigned.

The Otsu binarization method created by Nobuyuki Otsu assumes that the image to be thresholded contains two classes of pixels:

* foreground (something with interest for the user) and
* background (no interest for the user). It calculates the optimum threshold separating those two classes so that their combined spread (intra-class variance) is minimal (this represents the error).

The main advantage is the speed because we only need to compute the histograms and arrays of 256 and the easiness of the implementation.

# Architectural Design

In this chapter we will present the overall architecture along with the architecture design for each major system component.

## 5.1 System Architecture

The solution as a whole is taken into consideration. The complete system is made out of 2 large components:

* the Binarization Software Module and
* the Voting Module

The Voting Module should be able to receive and process the information from multiple Binarization Software components which can be deployed and launched on different machines.

For this general view of the entire system, we propose a Client-Server architecture:

* the server side will be represented by the Voting System ,
* while the Binarization Software component should represent the client part

The Voting component should be a centralized system. It should be up and running and should be able to receive and process connections with various Binarization Software components. It should also contain a database as to store the results generated by each of the individual Binarization modules as to use them for further analysis.

When launched the Binarization application components should try and connect to the Voting System by issuing a connection request and waiting for an acknowledgement from the server. In the case that no reply is received, a timeout is generated, meaning that a connection could not be successfully established.

Upon finishing the image processing, the Binarization application component should send the status history along with the corresponding binarized images to the Voting-System. When the information is received, the servershould send back an acknowledgement to the BAM component, letting it know that data transfer was successful. Only, after receiving the acknowledgement form the Voting-System, the BAM application should proceed with clearing the status history.

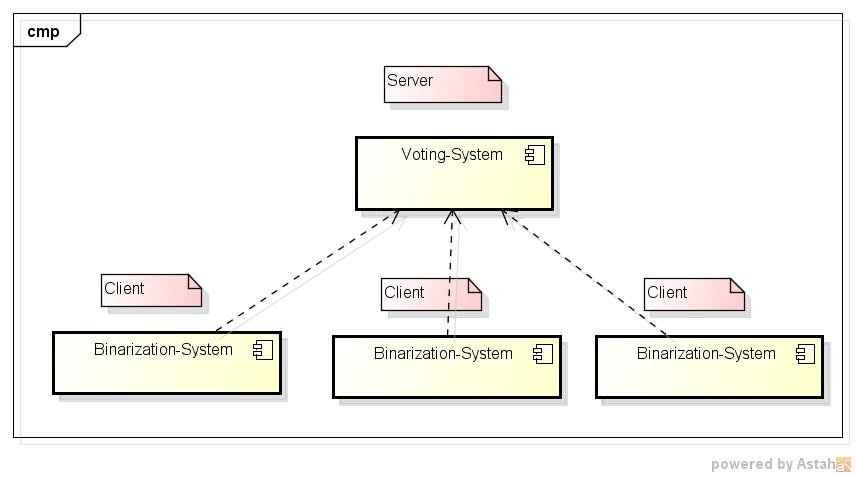


Figure System Architecture Design Overview

## 5.2 Architectural Patterns

In this section we will further describe the architectural details of each module.

The solution we have chosen for the Binarization module is based on a Layered Architecture. We recommend a 2 layered architecture:

* **Presentation Layer** – This is the topmost level of the application. It communicates with other layers by outputting results to the client layer and all other layers in the network. It is the user interface. The main functions of the interface are to translate tasks and results to something the user can understand.
* **Application Layer (Business logic/Logic Layer/Data Access Layer/ Middle Layer)** – The logical layer is pulled out from the presentation layer and as its own layer.It controls an application’s functionality by performing detailed processing. This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data with the Presentation Layer.

This solution offers the necessary decoupling between the graphical elements used within the graphical user interface and the corresponding controls that cover the actual processing and the business logic of the software.

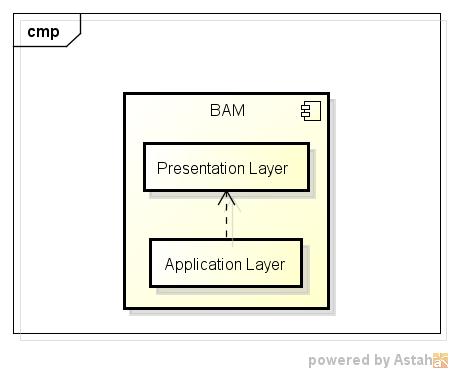


Figure BAM Individual Architecture Design

For the Voting Module we would recommend a 3-tier architecture containing the following layers:

* **Client layer:which** contains the functionality of connecting to the Binarization modules. This layer is used to define the way in which the module communicates with the other components from within the system. This layer offers the solution for taking the necessary input form the BAMs and communicating the computed results back to them.
* **Business layer:**In this layer all business logic is written (validation of data, calculations, data insertion etc). This acts as an interface between the Client layer and Data Access Layer. This layer is helps to make communication faster between client and data layer.
* **Data layer:**In this layer the actual database is comes in the picture. The Data Access Layer contains methods to connect with the database and to perform insert, update, delete, get data functions from database based on our input data.



Figure VBAM Individual Architecture Design

## 5.3 Implementation Requirements

To binarize images, we will use Otsu’s method, which is a very clever solution created by Nobuyuki Otsu. The algorithm assumes that the image to be threshold contains two classes of pixels or bi-modal histogram (e.g. foreground and background) then calculates the optimum threshold separating those two classes so that their combined spread (intra-class variance) is minimal

.

Its main advantages are speed (because we only use histograms and arrays of length 256) and the easiness of the implementation.

In addition, theVoting Binarization Algorithm Module, will usea “smart-voting” technology in order to blend the independent BAMs results into a binary image.

## 5.4 Components Interaction

Here is the overall component diagram that shows both the wy in which the components interract along with the architerural design of each individual module.

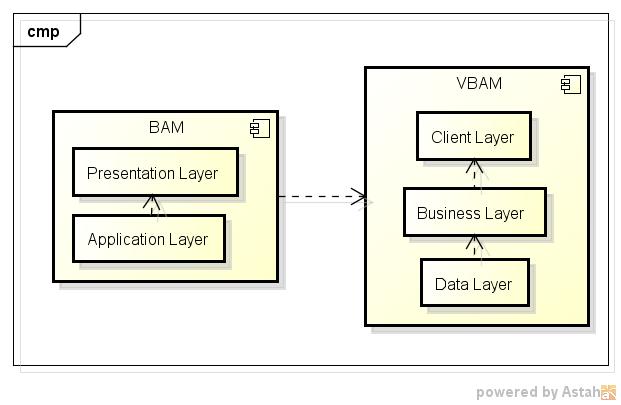


Figure System Component Diagram

# User Interface Design

The BAM-solution has a user friendly design, it is easy to use and accommodate with. The graphical interface is designed both for professional and personal use. The user interface should offer the following services:

* Browse for desired image. Only supported formats are displayed.
* Select the directory in which the binarized image should be saved.
* Choose the image format under which the result should be saved.
* Display the result.
* Choose settings.

## 6.1 Flow Charts

1. **Set Input Format**

The user can set which input formats should the application be looking for when browsing for images. Based on these settings the application will create a filter that will be applied in the browsing window.

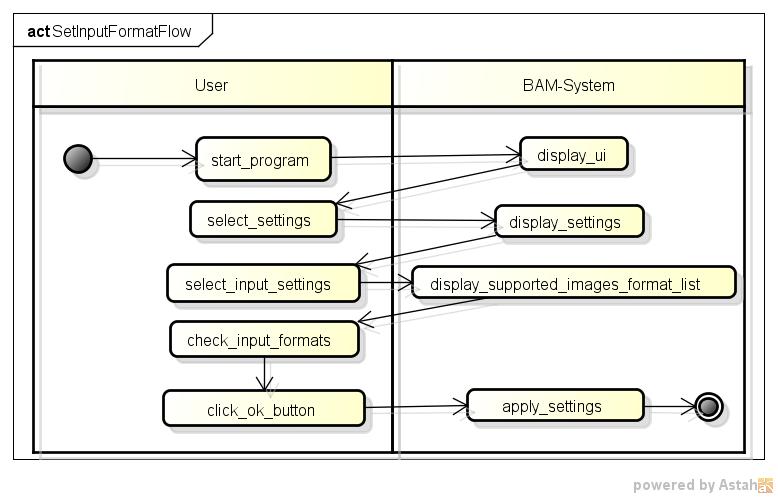


Figure Set Input Format Flow

1. **Select Output Directory**

The user can specify where the output image should be placed after it is created by the application.

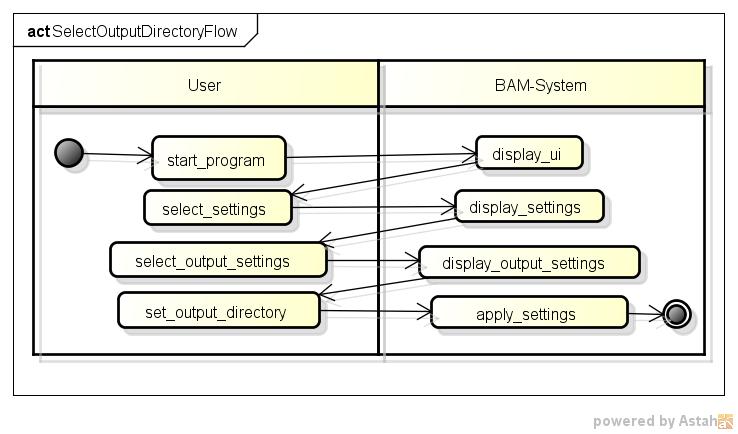


Figure Select Output Directory Flow

1. **Select Output Format**

The user can define the output image type. Based on this information the application will save the resulted image under this specific format.

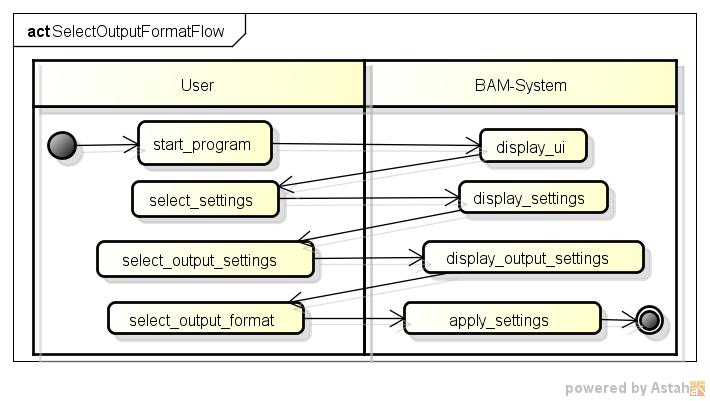


Figure Select Output Format Flow

1. **Browse For Image**

The user has to specify the input image. The application offers a browsing functionality and it is able to display only supported images. After the user has selected the input image the applications stores the choice and proceeds with the image processing. It then creates the output image based on the output settings and displays the results to the user. Once the output has been created, the BAM-System also informs the Voting-System of the outcome.

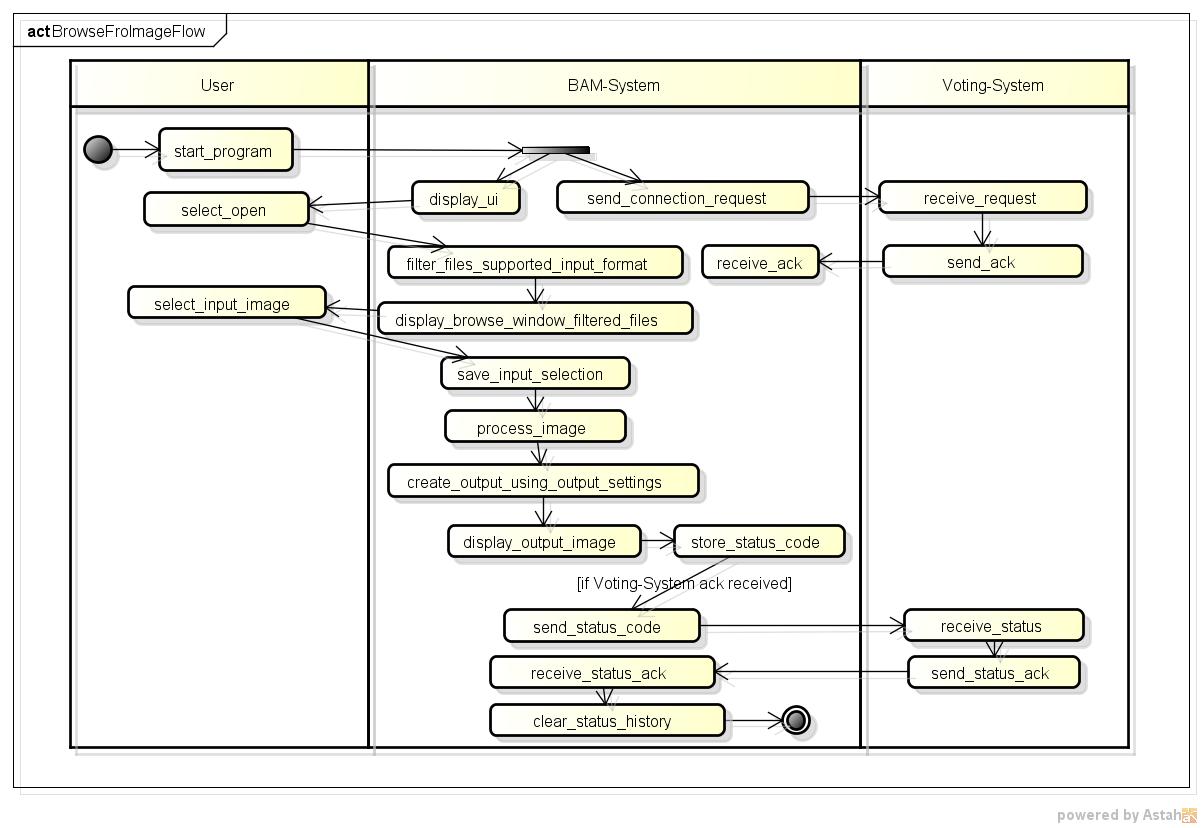


Figure Browse for Image Flow

# Testing Issues

## 7.1 Critical Components. Alternatives

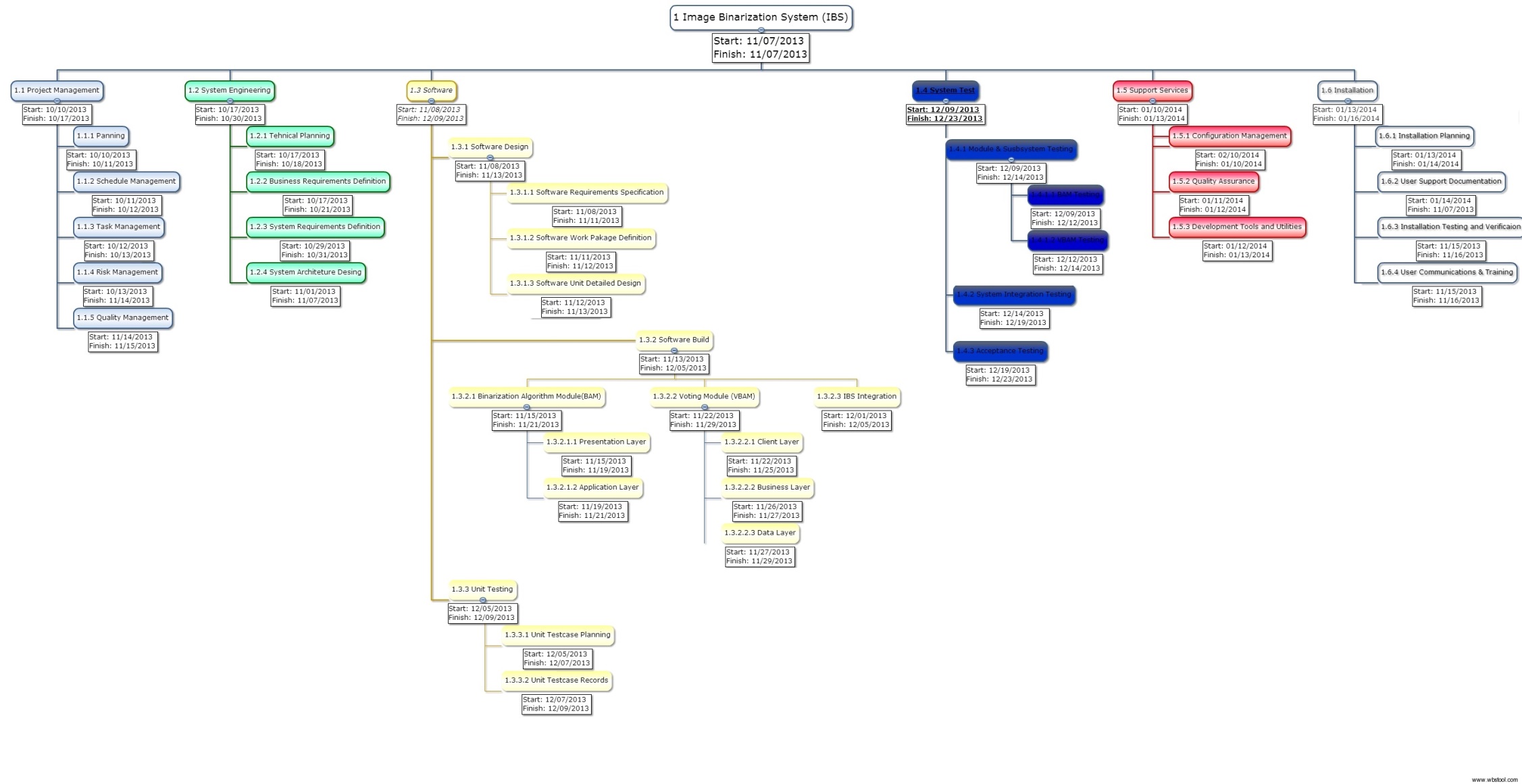
A possible issue may be related with the connectivity between the BAM-System and the Voting-System. If the connection fails, the BAM-System cannot inform the Voting component of the image processing outcome.

A persistent history of the outcomes should be stored internally by the BAM-System along with the binarized images. Upon sending the results to the Voting component, the BAM-System should clear the history. In this way, the solution handles the case in which the connection between the 2 modules may be lost. The information will be transmitted to the Voting-System once the connection is up and running again.

Another problem may occur while the Voting-System is in the process of sending the status-acknowledgement back to the BAM-System. In this case, the Voting-System has already stored and processed the information from the BAM-System, but the latter did not receive the notice. So, the BAM-System will not clear the status history and thus will try to resend it the next time around. As a solution, the Voting-Module should always check that the received data is not a duplicate of some piece of information that was already stored and processed.

# Work Breakdown Structure Diagram

The original image that offers a 100% scale can be found in the “WBS-IBS.jpg” file.



# Gantt Diagram

The Gantt chart can be found within the “Gant-IBS.pdf” file.

# Git Repository

Here is the Git repository that we have created for this project: <https://github.com/BINARYIMAGES/BAMOTSU>